UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en ingeniería informática

TRABAJO FIN DE GRADO

Artificial Intelligence Chef

Alejandro Albarca Molina
Tutor: Eduardo César Garrido Merchán
Ponente: Daniel Hernández Lobato

Junio 2018

Artificial Intelligence Chef

AUTOR: Alejandro Albarca Molina
TUTOR: Eduardo César Garrido Merchán

Dpto. Aprendizaje Automático
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio 2018

Resumen (castellano)

Este Trabajo Fin de Grado consiste en la creación de una herramienta que sea capaz de predecir cómo de bueno estará un plato de comida dependiendo de los ingredientes que tenga y cómo sean cocinados. Así mismo, el sistema también debe ser capaz de sugerir qué ingredientes usar, en qué cantidad, cuánto y cómo cocinarlos y cómo combinarlos de tal forma que se consiga cocinar ese plato de comida con un buen resultado final.

Con este trabajo, optamos a crear una herramienta que facilite la vida de las personas, y que en el futuro, con más recursos y con la ayuda de profesionales del mundo de la cocina, se pueda desarrollar hasta el punto de que cualquier usuario por poca experiencia que tenga, sea capaz de elaborar platos nutritivos, sanos y de buen sabor para su día a día.

En el presente documento se navegará a lo largo de todas las tecnologías que se han utilizado para implementar el chef de inteligencia artificial viendo cómo integrarlas para lograr nuestro objetivo.

Se tratará en todo momento de que el lector pueda entender el modo de proceder que se ha seguido aún sin tener grandes conocimientos del ámbito que se trata, viendo en primer lugar cómo se han recopilado los datos, para después ver cómo conseguir un modelo que sea capaz de predecir cuan bueno será un plato y finalizar usando una herramienta de optimización que nos sugiera qué hacer para cocinar el plato que solicitemos. Para terminar, puesto que este proyecto tiene una gran proyección y aún queda mucho potencial por explotar, veremos qué es necesario de cara al futuro para poder lograr resultados profesionales y útiles a gran escala.

Palabras clave (castellano)

Inteligencia artificial, optimización Bayesiana, búsqueda en rejilla, cocina

Abstract (English)

This Bachelor Thesis consists of the creation of a tool that will be capable of predicting how nice a meal would be depending on the ingredients it contains and how they are cooked. At the same time, the system will be able to suggest what ingredients can be used, the amount of each of them, how to Cook them and how much time, and finally how to combine them in a way that the final dish is prepared with a perfect final result.

With this project, we choose to create a tool that eases people's lives, and that in the future, with some more means and the help of professionals from the cooking world, this can be developed so that any user, even with little experience, could be able to prepare nutritious, healthy and flavourous dishes for the day to day.

In the present document we will navigate throughout all the technologies that have been used to implement the artificial intelligence chef watching how to integrate them in our project to achieve our aim.

In every moment, we will try that the reader can understand the procedure that has been followed even without having great knowledge of the field that will be considered, seeing first how the data has been collected, for later observing how to get a model that is able to predict how good a dish will be, and end using an optimization tool that suggests what to do to cook the meal we request. Finally, since this project has a great projection and there is still a lot of potential to be exploited, we will see what is needed facing the future to be able to achieve professional results and useful on a large scale.

Keywords (English)

Artificial intelligence, Bayesian optimization, grid search, cooking

Agradecimientos

En primer lugar me gustaría dedicar este trabajo de fin de grado a toda mi familia, que tanto me ha apoyado y ha creído en mí cuando ni yo mismo lo hacía, y tantas veces me han aguantado en esas épocas de exámenes en las que los estudiantes nos volvemos otras personas.

Especialmente me gustaría mencionar a mis padres. Nunca podré devolveros todo lo que habéis hecho por mí. Este grado es en gran parte vuestro, el esfuerzo que habéis hecho es incalculable, tanto mental como económicamente. Muchas gracias por inculcarme los valores que me han convertido en la persona que soy hoy. Nunca olvidaré esas primeras semanas de universidad en las que me veía incapaz de sacar esto adelante, y me enseñasteis una lección que no olvidaré nunca: con esfuerzo y fijándose objetivos pequeños, todo se logra.

No me olvido de aquel que me vio comenzar pero no me ha visto terminar. Abuelo, espero que te pudieras sentir orgulloso de mi.

Como no, gracias también a ti Samai, porque le has dado a todo esto una meta mucho más bonita y gratificante, ya sabes el objetivo por el que luchamos.

Gracias también a todos los profesores que me han formado a lo largo de este viaje, especialmente a mi tutor en este trabajo, Eduardo, sin su ayuda esto no hubiera sido posible. La disposición a tutorías y a resolver dudas ha sido inmejorable, ya fuera por correo, por Skype o de forma presencial, gracias a lo cual, he podido disfrutar de cada hito que íbamos logrando a lo largo de la realización del proyecto.

Y finalmente gracias también a ti, lector, por dedicar una parte de tu preciado tiempo para leer este trabajo. Espero que puedas disfrutarlo y que te lleves algo positivo de lo que se expone.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

En esta sección se explica el motivo de la realización del proyecto, así como el enfoque adoptado a la hora de hacer el trabajo. Finalmente, explicaremos cómo se ha estructurado la memoria.

1.1 Motivación

A lo largo de toda la historia la cocina siempre ha estado presente. Desde el comienzo de la humanidad hasta hoy día, la alimentación ha sido y es una parte fundamental de nuestra vida, y una clara prueba de ello es que las técnicas culinarias están en constante evolución.

De manera simultánea, el campo de la inteligencia artificial ha sufrido un desarrollo exponencial, lo cual ha permitido que el hombre pueda liberarse de trabajos que antes eran manuales para dedicarse a otras tareas más complejas. Gracias a esto, ahora es posible tener sistemas que permitan que nuestro vehículo se conduzca solo, aplicaciones que nos enseñen cuál es la mejor forma de estudiar un idioma, o sencillamente robots que trabajen en cadenas de montaje de una forma más rápida y exacta a como lo haría un ser humano.

A pesar de estos avances, el campo de la cocina no ha tenido grandes ayudas al respecto. Pocos son los sistemas creados para ayudar a que nos sea más fácil preparar platos sabrosos sin grandes conocimientos, pudiendo resumirlos en robots que nos indiquen qué alimentos necesitamos para una determinada receta y en qué orden debemos añadirlos para que se cocinen correctamente.

En este marco, es de gran necesidad y utilidad la investigación para poder crear platos de comida innovadores, que sean a priori inconcebibles para el hombre, a pesar de formarse en base a conocimiento experto previo basado en el conocimiento del humano.

Es por ello, que la principal motivación de este trabajo sea la de crear con unos recursos limitados, y de manera experimental, una herramienta que permita evolucionar platos de comida muy sencillos, de tal forma que nos recomiende recetas que en un primer momento no hubiéramos sido capaces de inventar, viendo así si los resultados son satisfactorios para poder extrapolarlos a platos de mayor complejidad y con una mayor cantidad de recursos.

1.2 Objetivos

El objetivo de este proyecto es la creación de una herramienta que en base a un conjunto de datos obtenido de manera experimental de un plato de comida, sea capaz de recomendar una receta para dicho plato de comida haciendo una combinación de los alimentos y tiempos de cocina que han obtenido una mejor puntuación en cada uno de los experimentos.

De este modo, la aplicación podrá ser utilizada tanto de manera profesional, pues permitirá a expertos del mundo culinario la elaboración de recetas que sin la ayuda del sistema quizás no hubieran sido capaces de inventar, como fuera del ámbito profesional, ya que ayudará a estudiantes, jóvenes o cualquiera persona con poco tiempo, a realizar comidas sabrosas sin tener grandes conocimientos de cocina.

A nivel académico, los objetivos de este proyecto se han centrado en poder desarrollar las habilidades relacionadas con el aprendizaje automático, teniendo que desplegar los conocimientos adquiridos en asignaturas como *Inteligencia artificial* o *Fundamentos de aprendizaje automático*. Así, se ha tratado de adquirir destreza con el lenguaje de programación *Python*, especialmente con la librería *sklearn* y herramientas como *Spearmint*. También se ha tratado en todo momento de tener un conocimiento a nivel matemático básico de lo que se ha realizado.

1.3 Organización de la memoria

Para facilitar el seguimiento lógico del desarrollo del proyecto, se ha estructurado la memoria de tal forma que sea sencillo saber cómo se ha realizado cada sección del mismo. Cada capítulo se basa en los experimentos logrados en el capítulo anterior, por lo que se puede saber cuál ha sido la cronología de la realización del trabajo de tal forma que si alguien quisiera reproducir el proyecto de manera independiente, sea capaz de llevarlo a cabo de manera satisfactoria. Las secciones en las que se diferencia la memoria son las siguientes:

- **Capítulo 1: Introducción.** Este es el capítulo actual, donde se ha podido ver la necesidad de la realización de un proyecto de investigación para la elaboración de una herramienta que ayudará a hacer más sencilla la vida de muchas personas.

- **Capítulo 2: Estado del arte.** Sección en la que se verá de manera global el punto desde el que partimos para saber qué avances hay ya hechos en el campo del aprendizaje automático respecto la cocina. También se verán algunas de las herramientas usadas en la realización del trabajo.

- **Capítulo 3: Diseño.** Capítulo donde se describe los distintos componentes que han compuesto la herramienta para el análisis y optimización de los platos.

- **Capítulo 4: Desarrollo.** Aquí se explica cómo ha sido el desarrollo del proyecto, los pasos que se han dado, las dificultades encontradas, y las soluciones aplicadas.

- **Capítulo 5: Experimentos realizados y resultados.** En este capítulo se verán cuáles han sido los resultados de los experimentos que se han hecho.

- **Capítulo 6: Conclusiones y trabajo futuro.** Finalmente analizaremos los resultados obtenidos, viendo si eran los que esperábamos, si tienen sentido, a qué se deben, y cómo poder mejorar la herramienta en el futuro.

2 Estado del arte

Este trabajo de fin de grado está directamente relacionado con el aprendizaje automático. En esta sección veremos los campos concretos que nos incumben del aprendizaje automático, para poder analizar en qué punto nos encontramos y hacia donde nos dirigimos, pues es vital conocer antes de empezar, cuál es el objetivo real a lograr con el presente trabajo.

2.1 Aprendizaje automático

El aprendizaje automático [1] es un área de la inteligencia artificial que versa sobre la modelización de funciones desconocidas de las cuáles se conocen datos que responden al comportamiento de dichas funciones. Cada uno de estos datos puede tener asociados o no una etiqueta, clase o número entre otros datos que representan al resultado de la función desconocida. En función de estas asociaciones encontramos las distintas subáreas del aprendizaje automático.

La evolución del aprendizaje automático está llevando a que cada vez haya un mayor número de modelos más complejos y flexibles, lo cual hace que tengan un número mayor de parámetros, implicando un aumento directamente proporcional de la complejidad y coste computacional de la algoritmia. Sin embargo se quiere lograr que se mantengan unos procedimientos de ajuste exactos, pues de ello puede depender el éxito o fracaso de importantes decisiones.

Actualmente existen gran variedad de campos en los cuales se pueden aplicar los beneficios del aprendizaje automático, como la medicina, o la interpretación del lenguaje y también son numerosos los algoritmos que se pueden aplicar para lograr los objetivos que se propongan. Una de estas áreas es la cocina, que es el tema que nos atañe en el presente documento, y del cual analizaremos el nivel de aprovechamiento que se hace del mismo en la actualidad.

Muchos de estos algoritmos basan su eficiencia en el análisis de la relevancia que tienen sus parámetros, para poder determinar cuáles de ellos merecen ser estudiados y optimizados y cuales no tienen gran impacto.

Existen numerosos mecanismos de búsqueda, por lo que en esta sección veremos:

- Cuál es el estado actual de la optimización Bayesiana
- Qué herramientas tenemos que lo implementen
- El estado del aprendizaje automático
- El estado de la cocina automática

2.1.1 Optimización Bayesiana

Es difícil establecer una fecha exacta en la que apareció la optimización Bayesiana. Este método está estrechamente relacionado con las ideas estadísticas para un diseño óptimo de experimentos que relate Kirstine Smith [2] en el año 1918, pero no fue hasta el año 1964 cuando Kushner [3] lo estudió bajo el nombre de optimización Bayesiana.

No fue hasta el año 2007 cuando se le dio atención con la conferencia NIPS (Neural Information Processing Systems) llevada a cabo por Brochu Eric, Nando de Freitas y Abhijeet Ghosh [4]

Supongamos que tenemos una función de la cual queremos saber su valor máximo pero no tenemos ninguna información acerca de ella. Las funciones de optimización son como una *caja negra* a la cual le pasamos esta función y nos devuelven el punto que estamos buscando. Es aquí donde nos es de utilidad el método que estamos estudiando, pues la optimización Bayesiana permite optimizar funciones cuya expresión analítica no se conoce y que son ruidosas, por lo que la evaluación de la función a la que se desee llegar es costosa en términos de tiempo; por otra parte, aún sin tener en cuenta el coste en tiempo, la actual computación en la nube puede ser determinante para saber también el coste económico, pues por lo general los problemas de aprendizaje automático necesitan correr en múltiples núcleos simultáneamente.

La optimización Bayesiana es un mapeo entre dos tipos de problemas, pues trata de modificar el problema de búsqueda de hiperparámetros que históricamente se ha resuelto tratando de minimizar una función conocida en un punto, como muestra la expresión (1)

$$x_M = \arg\min_{x \in \mathcal{X}} f(x) \quad (1)$$

en una serie de problemas menores buscando el punto siguiente a evaluar que maximice el valor de la función que se busca y que es desconocida.

Este método trabaja asumiendo que una función desconocida a la que queremos llegar procede de un proceso Gaussiano, entre otros modelos sin pérdida de generalidad, para posteriormente hacer experimentos probando con diferentes hiperparámetros para ver cuál es la mejor forma de aproximar la función a estudiar.

Hay tareas no muy complejas que pueden ser resueltas con optimización Bayesiana, como la clasificación de imágenes o el reconocimiento de diferentes lenguajes. Pero aparte de estos, el estado del arte está avanzando en una dirección mayor, pues puede ayudar a encontrar nuevos materiales orgánicos que podrían servir para construir nuevas pantallas para teléfonos inteligentes, ayudar a optimizar el sistema de control de robots como los realizados en el *MIT* (*Massachusetts Institute of Technology*) o el diseño de nuevas afinidades de proteínas como las realizadas por el *Broad Institue*.

Por tanto, podemos decir que el campo de la optimización Bayesiana tiene un gran potencial de explotación y de hecho hay grandes áreas de desarrollo al respecto a pesar de ser muy joven pues su estudio real es muy reciente.

2.1.2 Búsqueda en rejilla

La búsqueda en rejilla [5] o *grid search*, es un método que nos conviene conocer en el área de la optimización Bayesiana pues tradicionalmente ha sido la forma más común de realizar la búsqueda de los hiperparámetros óptimos.

La búsqueda en rejilla consiste en una búsqueda minuciosa de manera manual de una serie de hiperparámetros de un espacio del algoritmo de aprendizaje que estemos estudiando, analizando qué combinación de todas las posibles de los hiperparámetros es la que nos aporta los mejores resultados.

La forma de analizar cuáles de estos resultados son los mejores, típicamente se realiza mediante una métrica de rendimiento que suele ser la validación cruzada sobre el conjunto de entrenamiento.

En algunas ocasiones, los parámetros de nuestro sistema pueden ser valores reales limitados pero sin embargo también pueden ser ilimitados, por lo que en estos casos es necesario establecer ciertos límites y discretizar antes de realizar la búsqueda. Por tanto, de esto podemos concluir que este tipo de búsqueda puede llegar a sufrir la maldición de la dimensionalidad.

Para entender mejor cómo funciona la búsqueda en rejilla pongamos un ejemplo. Supongamos un sistema de aprendizaje automático que usa el algoritmo de vecinos próximos (k-nn) y que puede usar dos métricas para medir la distancia: la distancia euclídea o la distancia Manhattan. Con esto, se formaría una rejilla como la que se puede observar en la figura 2.1.2-1. En uno de los ejes se establecen los distintos valores de los vecinos próximos que tomaremos, y en el otro eje cada una de las dos medidas, de tal forma que los puntos de corte de unos y otros nos indican los puntos con los que probaremos y estudiaremos su eficacia. De este modo, veríamos cuál de ellos nos da el resultado óptimo, que en este ejemplo se ha marcado con un punto verde correspondiente a 5 vecinos con la distancia Manhattan.

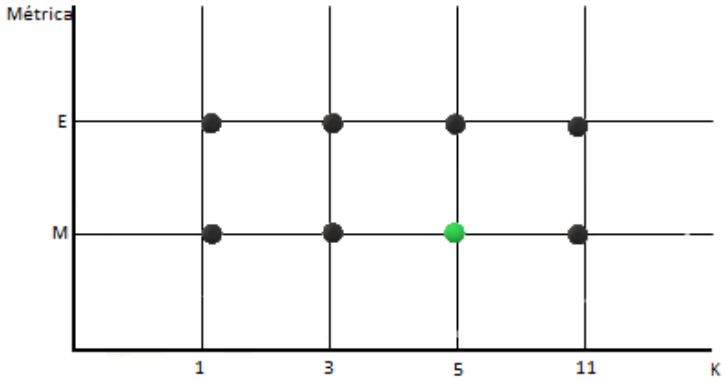


Figura 2.1.2-1: Ejemplo de búsqueda en rejilla

Este tipo de búsqueda se puede realizar no sólo con 2 parámetros, sino también con varios. En las figuras 2.1.2-2a y 2.1.2-2b se puede ver la comparación de cómo sería con 2 y 3 parámetros.

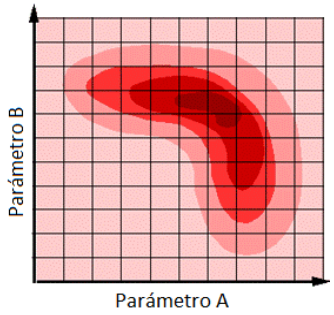


Figura 2.1.2-2a: Búsqueda 2 parámetros

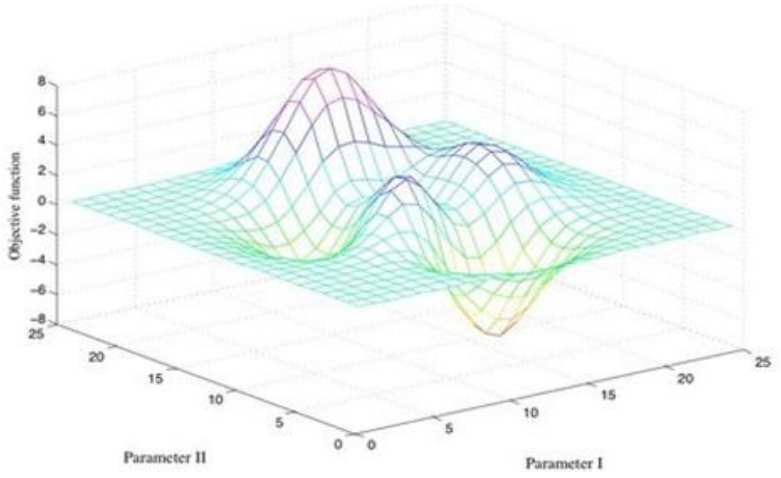


Figura 2.1.2-2b: Búsqueda 3 parámetros

2.1.3 Búsqueda aleatoria

Como hemos mencionado anteriormente, la búsqueda en rejilla al ser un método tan exhaustivo puede adolecer de la maldición de la dimensionalidad y por tanto llegar a ser costoso computacionalmente, y si no se realiza en paralelo, también en términos de tiempo. Es aquí donde nace la idea de la búsqueda aleatoria [6].

Este tipo de búsqueda sencillamente muestrea una combinación de parámetros elegidos de manera aleatoria un número fijo de veces. El potencial de este tipo de búsqueda frente a la búsqueda en rejilla aparece en espacios de una alta dimensión.

En las figuras 2.3.1-a y 2.3.1-b podemos ver la comparación de la búsqueda en rejilla con la búsqueda aleatoria. Como se observa, la búsqueda en rejilla examina sin excepción cada una de las opciones que hay de manera organizada, mientras que la búsqueda aleatoria selecciona conjuntos de estudio al azar.

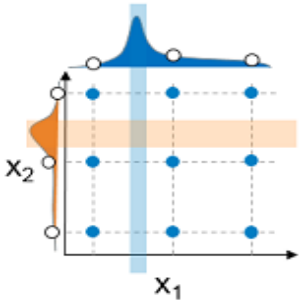


Figura 2.1.3-1a: Búsqueda en rejilla

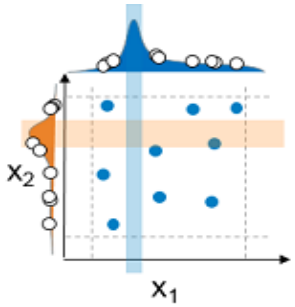


Figura 2.1.3-1b: Búsqueda aleatoria

Existen además otros métodos para la búsqueda óptima de parámetros, como es el caso de los algoritmos genéticos.

2.1.4 Metaheurísticas: Algoritmos genéticos

Este tipo de algoritmos [7] se basan en la teoría de la evolución por lo que siguen el método de prueba y error.
Hablando en los términos de la naturaleza, podríamos hacer la siguiente analogía:

- Hay una población de individuos que luchan por unos recursos
- Los individuos que mejor se adaptan son los que sobreviven
- Ahora, en un algoritmo genético:

    - Problema → entorno
    - Función de ajuste → mide la adaptación al entorno
    - Conjunto de soluciones → población
    - Cada individuo → posible solución

- Mecanismos de adaptación aleatorios → cruce, selección, mutación

En esencia, un algoritmo genético trabaja como sigue:

1º Se crea una población aleatoria P
2º Mientras no se cumpla una condición de terminación:
P' = selección de progenitores (P)
P' = recombinación (P)
P' = mutación (P')
P' = selección mejor (P, P')
3º Devolver (P')

Lo primero que se realiza es una estructura de datos que sea capaz de admitir las posibles soluciones a un problema sabiendo que cada una de estas posibles soluciones será un *individuo* hablando en términos evolutivos. Como existirán algunas soluciones mejores que otras, en cada iteración del algoritmo se seleccionarán las N mejores soluciones, y el número de soluciones restantes se completará con nuevas soluciones que contendrán las partes más aptas de generaciones pasadas.

Así, la manera de proceder es combinando los mejores individuos entre sí, tratando que intercambien información estructurada de una forma pseudoaleatoria.

Ejemplifiquémoslo. Imaginemos que tenemos un conjunto de datos con 20 atributos más una clase, y queremos seleccionar el subconjunto de esos atributos que hace la clasificación óptima mejorando el rendimiento computacional. En primera instancia se debe crear una población de N individuos aleatorios, de tal forma que cada individuo sea un conjunto de 20 bits, donde un bit a 1 indique que sí tendremos en cuenta ese atributo para la clasificación, y un bit a 0 indique que no será tenido en cuenta. Con cada uno de esos individuos, clasificaremos en el conjunto de entrenamiento de nuestro conjunto de datos y obtendremos un porcentaje de acierto, que indique la valía de ese individuo. Nos quedaríamos con los M individuos con mejor porcentaje de acierto y los seleccionaríamos para la siguiente generación. Los N-M individuos restantes saldrían del cruce y mutación de los mejores individuos de generaciones anteriores. De este modo, al finalizar las iteraciones, habrán sobrevivido los mejores individuos, y tendremos nuestros hiperparámetros seleccionados de una forma óptima.

Como hemos visto, hay numerosos algoritmos para la búsqueda de hiperparámetros, siendo la optimización Bayesiana uno de los más novedosos. En la siguiente sección analizaremos algunas herramientas para poder aplicar este algoritmo.

2.2 Herramientas de optimización Bayesiana

Ya hay varias librerías de código abierto que permiten de una manera sencilla aplicar optimización Bayesiana. Veamos dos ejemplos de ello

2.2.1 Spearmint

Speartmint [8] es un paquete gratuito para uso académico de software creado para realizar optimización Bayesiana. El software está creado para automáticamente ejecutar los experimentos que queramos realizar de tal manera que iterativamente se ajusten los parámetros para minimizar los objetivos que se propongan en el menor número de iteraciones posibles.

Generalmente en la optimización Bayesiana hay dos decisiones importantes a tomar:

1 Elegir una función que sepa decidir en base al pasado las suposiciones que se harán en el futuro (se elige un proceso gaussiano (GP) debido a su flexibilidad y trazabilidad)

2 Crear una función de adquisición que construya la función para un modelo posterior que nos ayudará a elegir cuál es el siguiente el punto a evaluar

Existen varios criterios a utilizar para este fin. Este paquete lo realiza maximizando el valor que se espera que vaya a tener una mayor mejora alrededor del valor actual.

2.2.2 rBayesianOptimization

Otra alternativa para utilizar la optimización Bayesiana es el paquete rBayesianOptimization [9] que en este caso está implementado en lenguaje R.

El uso de la misma es muy sencillo, tan solo es necesario llamar a la función *BayesianOptimization()* que recibe una serie de parámetros entre los que destacamos

- **FUN:** es la función a optimizar
- **Bounds:** lista con los valores alto y bajo de cada hiperparámetro
- **N_iter:** número de veces que se repetirá la optimización Bayesiana
- **Acq:** donde podemos elegir el tipo de la función de adquisición

Finalmente, esta función nos devuelve

- **Best_Par:** vector de los mejores hiperparámetros encontrados
- **Best_Value:** el valor de las métricas logradas por cada hiperparámetro
- **Histroy:** tabla de datos del historial de la optimización Bayesiana
- **Pred:** tabla de datos tras aplicar validación cruzadaa a cada ronda de la optimización Bayesiana

2.3 Aprendizaje automático y su aplicación en la cocina

En la actualidad ya hay grandes compañías que han comenzado a aplicar el aprendizaje automático en el mundo de la cocina. En esta sección veremos dos ejemplos de ello, analizando qué han hecho y los resultados que han obtenido

2.3.1 Chef Watson

Chef Watson [10] es una herramienta creada por IBM que es capaz de ayudar a los cocineros a descubrir e inventar recetas culinarias mediante el análisis del sabor de unas recetas previas analizadas por algoritmos.

La manera de proceder de este sistema se basa en ingredientes; en primer lugar se pide un ingrediente que se desea que contenga la receta o al contrario, un ingrediente que no se desee que tenga la receta, a partir de lo cual, genera una receta en función de diferentes parámetros, como por ejemplo si el plato es para comer o cenar, para un postre o plato principal, para una época del año calurosa o fría, etc. Con esto, el sistema busca en su base de datos recetas que cumplan las características que el usuario desea. Cuando el sistema ha analizado una serie de posibles recetas, se encarga de clasificarlas para el usuario en *clásicas* o *únicas*.

El sistema tiene un módulo de análisis de lenguaje natural que le permite conocer más de 9000 recetas que tiene en su base de datos para saber cuál de ellas se pueden ajustar a lo que quiere el usuario. De manera adicional, Chef Watson tiene la capacidad de crear nuevas recetas gracias a que el mundo de la cocina tiene una serie de pares de ingredientes que casan entre sí de manera correcta; el sistema toma esos pares, y relacionando unos con otros es capaz de crear grupos como estos pero de hasta 7 u 8 ingredientes, lo cual ha dado lugar a interesantes platos culinarios si bien es cierto que esto aún está en fase de desarrollo.

2.3.2 Google galletas

También Google está investigando acerca de la optimización Bayesiana en relación a la cocina. En este caso, se han centrado en el optimizar una receta para galletas [11] con chips de chocolate.

La manera de proceder que ha seguido Google con su experimento es similar al que hemos seguido nosotros en el nuestro pero en una escala mayor. En este caso, lo que hicieron fue coger 20 recetas de galletas de chocolate y cocinar unas 90 galletas por receta para darlas a probar y puntuarlas con un rango entre 0-5 y obtener la recomendación. A medida que se iba obteniendo dicha puntuación, el algoritmo que utilizaron (Vizier, que está basado en optmización Bayesiana), se encargaba de sugerirles nuevas posibles

variaciones de las recetas. Con estas recetas obtuvieron una lista de ingredientes posibles para realizar las galletas así como unos rangos de cantidades de los mismos.

El experimento de Google fue más allá pues decidieron estudiar si las recetas optimizadas eran iguales en diferentes regiones, por lo que realizaron un segundo experimento pero esta vez en Mountain View, California (el primer experimento fue en Pittsburgh).

Los resultados fueron interesantes pues las costumbres culinarias que hay en cada región se vieron reflejadas en las recetas optimizadas, ya que cada cada población tenía unas preferencias diferentes lo cual se influyó en los ingredientes como se puede ver en la tabla 2.3.2-1, si bien es cierto que Google puntualizó que no se pueden comparar exhaustivamente ambos experimentos pues en cada uno de ellos había unas condiciones diferentes.

	Harina (gr)	Chocolate (gr)	Bicarbonato de sodio (cucharada)	Sal (cucharada)	Cayena (cucharada)	Azúcar (gr)	Huevo(gr)	Mantequilla (gr)	Extracto de naranja (cucharada)	Extracto de vainilla (cucharada)
Pittsburgh	167	196	0.5	0.25	0.25	108	30	129	0.375	0.5
California	167	245	0.6	0.5	0.125	127	25.7	81.3	0.12	0.75

Tabla 2.3.2-1: Resultados galletas de google

3 Diseño

En esta sección veremos cómo se ha acotado el problema y posteriormente se explicará cuál ha sido la lógica implementada para abordarlo.

3.1 Definición del problema

En el trabajo de fin de grado elegido, vamos a tratar de construir un sistema que nos determine si unos determinados platos de comida están buenos o malos, es decir, que sea capaz de dar una evaluación de la calidad de una receta según los ingredientes empleados y herramientas utilizadas para cocinarlos.
Cada receta puede ser representada mediante un conjunto de variables que describa:

- La variable empleada para cada ingrediente o herramienta
- El rango de valores que puede tomar dicha variable
- El tipo de variable
    - Binaria
    - Real
    - Entera
    - Categórica

En este trabajo se han propuesto dos problemas que encajan dentro de la definición descrita. Para cada uno de ellos, vamos a tratar de definir:

- Los atributos que podremos variar
- El objetivo que mediremos para saber si nuestra receta resulta satisfactoria o no

A continuación definimos las dos recetas que evaluaremos.

Perrito caliente

El primero de los platos que se plantea es un perrito caliente. El motivo, es que se considera que puede dar muchos datos con los que hacer un análisis aunque a priori es un plato de baja complejidad.
Los atributos que se van a medir para este plato son los que siguen:

- Tiempo que se hace la salchicha. Variable entera [0,200] segundos
- Tiempo que se tiene tostando el pan de perrito. Variable entera [0,200] segundos
- Sitio en el que se cocina la salchicha. Variable categórica [sartén, micro ondas]
- Cantidad de kétchup que se le pone. Variable entera [0,8] cucharadas
- Cantidad de mayonesa que se le pone. Variable entera [0,8] cucharadas
- Cantidad de mostaza que se le pone. Variable entera [0,8] cucharadas

En este plato mediremos un objetivo:

- Sabor del perrito. Media ponderada de la evaluación entre 0 y 10 de 3 personas que prueban la receta.

$$Sabor = \frac{Nota_1 + Nota_2 + Nota_3}{3} \quad (2)$$

Ensalada César

El motivo de la selección de este plato es porque debido a los muchos ingredientes que tiene, da muchas opciones de análisis.

En este caso mediremos los siguientes atributos:

- Número de la vitro cerámica al hacer el pan. Variable entera [1,9]
- Tiempo que se tiene tostando el pan. Variable entera [5,51] segundos
- Número de la vitro cerámica al hacer el pollo. Variable entera [1,9]
- Tiempo que se tiene el pollo. Variable entera [1,200] segundos
- Marca de la salsa César que se compre. Variable categórica [Ybarra, Heinz]
- Marca de lechuga. Variable categórica [El gigante verde, Carrefour, Cogollos, Curro]

En este plato mediremos un objetivo:

- Sabor del perrito. Media ponderada de la evaluación entre 0 y 10 de 3 personas que prueban la receta.

$$Sabor = \frac{Nota_1 + Nota_2 + Nota_3}{3} \quad (3)$$

Se ha establecido que para poder realizar una correcta evaluación de un plato de manera correcta, se deberá realizar un mínimo de 30 veces para cubrir todo el espacio de atributos, variando cada uno de los atributos que hemos definido anteriormente.
Para la evaluación, se contará con 3 notas, correspondientes a tres personas diferentes que evaluarán los platos a lo largo del proyecto. Cada una de ellas dará una nota para el objetivo según lo que ya se ha establecido y explicado.

3.2 Flujo del sistema

Debido a que el tratamiento del dato es una parte esencial del proyecto, el problema lo hemos diferenciado en tres partes

1. Aumento de los datos
2. Entrenamiento de un modelo que se ajuste a los datos
3. Predicción de la nota de una receta en función de unos datos entrada y sugerencias de recetas

La herramienta para el aumento de ha sido implementada con la librería *numpy* [12] de *Python*. Esta librería es el paquete principal de *Python* para computación científica. *Numpy* provee *arrays* multidimensionales, derivadas, operaciones matemáticas y de lógica, transformaciones de Fourier y simulaciones aleatorias entre otras muchas opciones. Esta librería se ha usado para generar distintas distribuciones de probabilidad.

Para el entrenamiento del modelo, se ha utilizado el paquete *sklearn* de *Python*. Esta librería proporciona numerosos algoritmos de clasificación, regresión y *clustering*, como

SVN (*support vector machines*), *random forest* o vecinos próximos entre otros. De esta herramienta se ha usado SVR (*support vector regression*), para dar una puntuación a las recetas.

Finalmente, para la predicción se ha usado *Spearmint*, el cual ha sido analizado en el estado del arte.

En la figura 3.2-1 se muestra el flujo de la herramienta.

1. Partimos de una receta de comida
2. Dicha receta nos proporciona ingredientes y herramientas que serán nuestras variables
3. Con la definición de las variables, su tipo y sus posibles valores, tenemos un conjunto de datos inicial
4. Este conjunto de datos lo ampliamos gracias al aumentador de datos
5. Con el conjunto de datos ampliado, seleccionamos un modelo, lo entrenamos y vemos si es válido. En caso negativo, repetimos este paso hasta encontrar un modelo válido
6. Mediante *Spearmint* podemos aplicar optimización Bayesiana que nos recomendará un valor óptimo de las variables que definimos en el paso 2

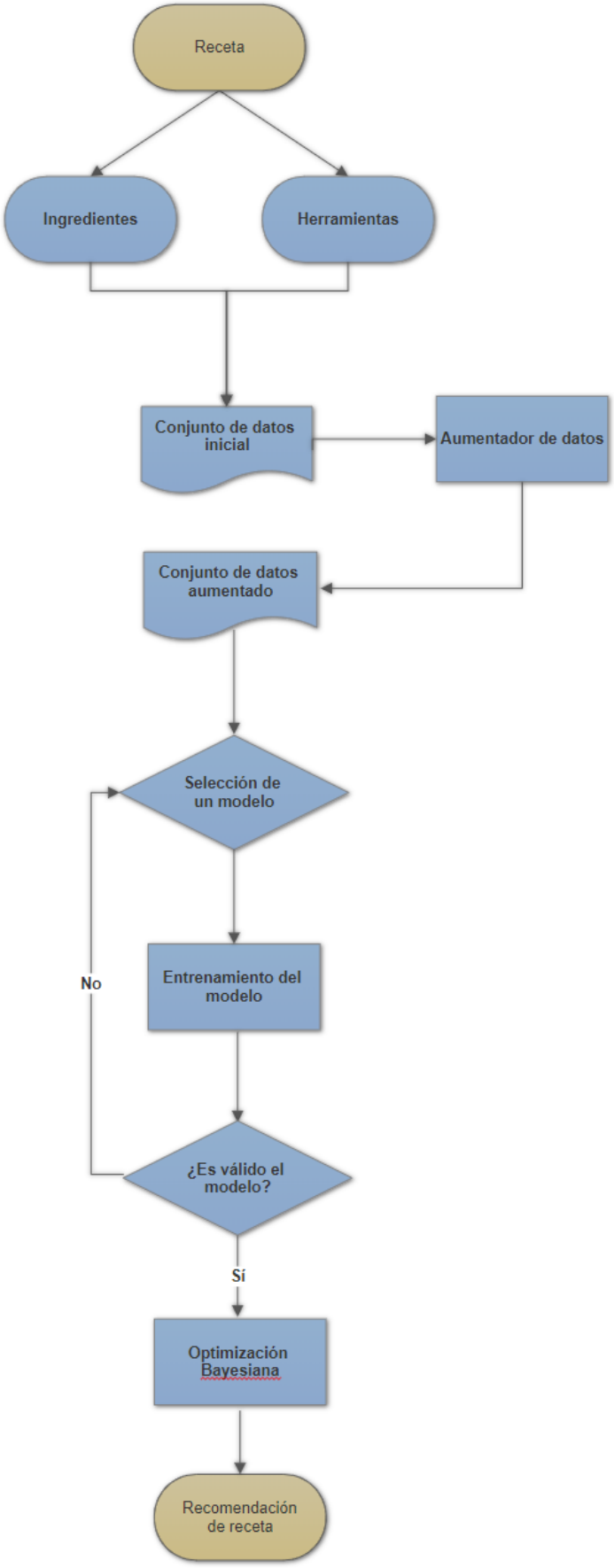


Figura 3.2-1: Diagrama de flujo de la aplicación

4 Desarrollo

En esta sección veremos cuál ha sido el procedimiento del proyecto, analizando los problemas que han surgido y viendo qué soluciones se han aplicado para su resolución.

4.1 Toma de datos

El primer paso para poder proceder, fue la toma de datos.
Para ello, se tomaron 44 registros de cada una de las dos comidas. Estos datos se recopilaron mediante 3 individuos que dieron sus notas e impresiones de manera independiente.
A la hora de la toma de datos, se solicitó a los individuos no sólo que dieran una nota a cada receta que probaran, sino que además trataran de dar la máxima cantidad posible de información, como por ejemplo si cierta combinación de salsas les gustaba, o el motivo de por qué un determinado plato obtuviera una nota especialmente alta o baja. Con esto se lograron una serie de reglas en lenguaje natural basadas en conocimiento experto, las cuales mostramos a continuación.

Perrito caliente:

- El mejor punto del pan se tiene con tiempo de [100,110] segundos
- La combinación de salsas que mejores resultados obtiene es aquella en la que hay 2 cucharadas de mayonesa y 1 de salsa barbacoa y mostaza, es decir, 4 cucharadas en total. También ha tenido buena puntuación 3 cucharadas omitiendo la salsa barbacoa o la mostaza, es decir 2 de mayonesa, 1 de salsa barbacoa y 0 de mostaza por ejemplo. Sin embargo, no es buena idea poner más de 4 cucharadas de salsas al perito, sea la combinación que sea, pues de ese modo queda demasiado cargado.
- A la hora de elegir entre sartén y microondas, la sartén obtiene con grandes diferencias una mejor puntuación pues la salchicha queda más jugosa y permite realzar el sabor de la carne. En la sartén, el rango de tiempo que consigue un buen sabor es de unos [35,45] segundos. En el microondas, el tiempo óptimo es de 70 segundos, con menos tiempo queda cruda y con más se reseca, si bien es cierto que como se ha dicho anteriormente, el microondas no tiene muy buenos resultados de ninguna de las formas.

Ensalada César:

- El pan se ha podido comprobar que el punto en el que mejor está es con la vitro cerámica al 9 y dejándolo unos [10,15] segundos. También se puede hacer con la vitro cerámica a menor temperatura y dejándolo más tiempo, pero tiene peores resultados debido a que absorbe más aceite.
- El mejor resultado para el pollo se logra con la vitro cerámica al [8,9] dejándolo entre [90,115] segundos.
- Las marcas de lechuga que más han gustado han sido la Carrefour y Cogollos.

- La salsa césar que más ha gustado en el conjunto final con la ensalada es la de marca Ybarra.

Por tanto, se logró que a la hora de aumentar los datos para el posterior análisis, no solo se hiciera mediante alguna técnica de repetición y reemplazamiento, sino que se hiciera en base a esas reglas que se recopilaron, de tal forma que se pudiera hacer un conjunto de datos lo más fiel posible a la realidad.

4.2 Aumento de los datos

El siguiente hito para poder continuar con el proyecto, consistía en aumentar la cantidad de datos. Idealmente, lo mejor hubiera sido que todos los datos con los que se hubiera realizado el entrenamiento del modelo fueran reales. Sin embargo, debido a que los recursos tanto en tiempo como en dinero han sido limitados, se ha procedido a realizar un aumento de los datos tratando de imitar de la manera más fidedigna posible la realidad.

Se trató de ver alguna herramienta o algoritmo que pudiera sernos de utilidad a la hora de hacer el aumento de datos. Para tal fin se estudió el *bootstrapping* [13] que es una técnica utilizada en aprendizaje automático cuando no se tienen suficientes datos para entrenar al algoritmo y se necesita ampliar el conjunto de datos. Esto se hace mediante un aumento uniforme de manera aleatoria seleccionando observaciones de los datos y duplicándolas con reemplazamiento.

Además, de forma adicional este método permite mejorar los resultados de los algoritmos, pues ayuda a reducir la varianza y evitar el sobre ajuste.

Sin embargo, esta técnica no nos aportaría ninguna información nueva relevante, por lo que se optó por realizar el aumento de los datos creando nosotros mismos un programa que lo implementara en base al conocimiento experto que obtuvimos.

Por ello, y para evitar realizar un modelo con sobreajuste, también se introdujo en la creación de los datos una componente aleatoria para simular el ruido que se tendría debido a los diferentes gustos de cada individuo.

Un aspecto importante en este punto, es tratar de cubrir todo el espacio de atributos, es decir, no sólo hay que generar datos para las combinaciones que dan una buena nota, sino que también tenemos que generarlos para las combinaciones que dan una mala nota para así asegurarnos de que el modelo sea capaz tanto de ver un buen caso como uno malo.

Así mismo, también es importante detallar que se ha observado que cada una de las reglas tiene valor independientemente del resto de variables que explican la variable sobre la cual se hace la regresión.

Perrito caliente

Tenemos tres componentes principales: el pan, la salchicha y las salsas. Por lo que tendríamos que generar datos para los siguientes casos:

Pan	Salchicha	Salsas
Mal	Mal	Mal
Mal	Mal	Bien
Mal	Bien	Mal
Mal	Bien	Bien
Bien	Mal	Mal
Bien	Mal	Bien
Bien	Bien	Mal
Bien	Bien	Bien

Tabla 4.2-1: Combinaciones conocidas para perrito caliente

Sin embargo, con esto estaríamos ignorando que no siempre vamos a saber cómo van a estar todos los ingredientes, por ejemplo, podemos saber que el pan está bien pero desconocer cómo estarán la salchicha y las salsas. Con esto, el espacio sobre el que trabajar aumenta, teniendo además que ver las suposiciones que muestran la tabla 4.2-2.

Pan	Salchicha	Salsas
Desconocido	Desconocido	Desconocido
Desconocido	Desconocido	Conocido
Desconocido	Conocido	Desconocido
Desconocido	Conocido	Conocido
Conocido	Desconocido	Desconocido
Conocido	Desconocido	Conocido
Conocido	Conocido	Desconocido
Conocido	Conocido	Conocido

Tabla 4.2-2: Combinaciones totales para perrito caliente

Esto es un aspecto necesario pero que a posteriori pudiera ser conflictivo, pues pudiera generar excesivo ruido. La solución por la que se optó fue que, a la hora de generar los datos, se introdujeran más registros de los experimentos en los que se sabe el valor de todas las variables, que de aquellos en los que se desconoce algo.

Ensalada César

Tenemos cuatro componentes principales: el pan, el pollo, la marca de la lechuga y la marca de la salsa césar. Con esto, tendríamos que cubrir los casos que indican la tabla 4.2-3

Pan	Pollo	Lechuga	Salsa
Mal	Mal	Mal	Mal
Mal	Mal	Mal	Bien
Mal	Mal	Bien	Mal
Mal	Mal	Bien	Bien
Mal	Bien	Mal	Mal
Mal	Bien	Mal	Bien
Mal	Bien	Bien	Mal
Mal	Bien	Bien	Bien
Bien	Mal	Mal	Mal
Bien	Mal	Mal	Bien
Bien	Mal	Bien	Mal
Bien	Mal	Bien	Bien
Bien	Bien	Mal	Mal
Bien	Bien	Mal	Bien
Bien	Bien	Bien	Mal
Bien	Bien	Bien	Bien

Tabla 4.2-3: Combinaciones conocidas para ensalada César

Aquí tenemos la misma problemática que antes, por lo que nos quedaría lo recogido en la tabla 4.2-4

Pan	Pollo	Lechuga	Salsa
Desconocido	Desconocido	Desconocido	Desconocido
Desconocido	Desconocido	Desconocido	Conocido
Desconocido	Desconocido	Conocido	Desconocido
Desconocido	Desconocido	Conocido	Conocido
Desconocido	Conocido	Desconocido	Desconocido
Desconocido	Conocido	Desconocido	Conocido
Desconocido	Conocido	Conocido	Desconocido
Desconocido	Conocido	Conocido	Conocido
Conocido	Desconocido	Desconocido	Desconocido
Conocido	Desconocido	Desconocido	Conocido
Conocido	Desconocido	Conocido	Desconocido
Conocido	Desconocido	Conocido	Conocido
Conocido	Conocido	Desconocido	Desconocido

Conocido	Conocido	Desconocido	Conocido
Conocido	Conocido	Conocido	Desconocido
Conocido	Conocido	Conocido	Conocido

Tabla 4.2-4: Combinaciones totales para ensalada César

Con el espacio de atributos definido, ya es posible comenzar con la implementación del generador de datos.

Para ello, se ha hecho uso de la librería *numpy* de *Python* pues permite el uso de una gran cantidad de distribuciones de probabilidad.

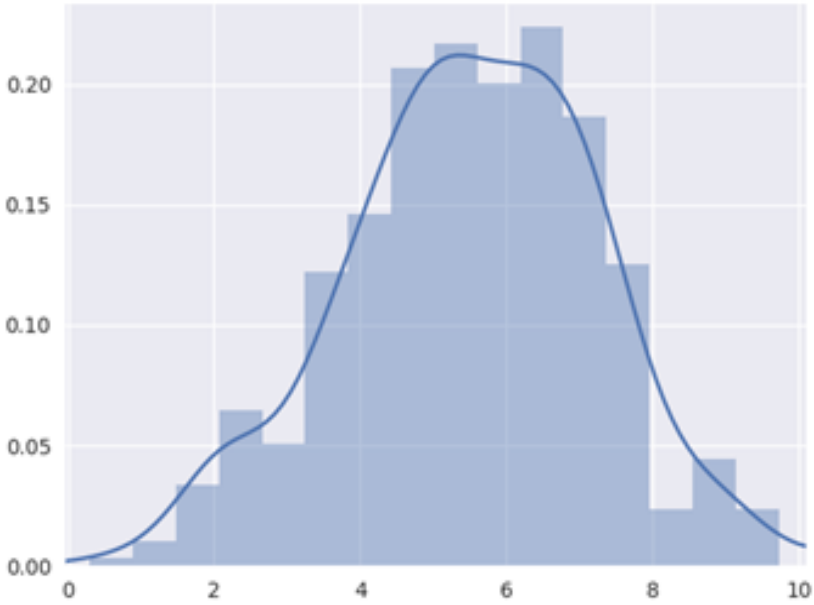
A la hora de hacer un aumento de los datos, el valor de cada una de las variables estaba acotada, pues es posteriormente la nota del plato lo que está condicionado en función de estos valores. Por tanto su generación se realizó con números aleatorios dentro de un rango que se preestablecía en cada caso. Estos rangos de valores, se obtuvieron en función de los datos recopilados en los experimentos reales y de las reglas de conocimiento experto.

El matiz realmente importante estaba en la nota, pues dependiendo de si los resultados del experimento eran buenos o malos, nos convendría una distribución de probabilidad u otra. Para los experimentos con una distribución de notas uniforme, se han usado distribuciones normales, mientras que para casos en los que se sabe que la nota será mala por los valores de los atributos, o que por el contrario fueran excepcionalmente buenas, se han usado otras distribuciones como la *chisquare*, la exponencial, la gamma o la distribución de *Poisson*. A continuación detallamos la expresión analítica de las distribuciones usadas, junto con la representación gráfica de la correspondiente distribución de probabilidad.

En el caso de la distribución normal, que precisa de la media y la desviación típica para poder realizarse, dichos parámetros se han calculado en base a cada ejemplo, es decir, para ver la media y la desviación típica del caso en el que el pan era una dato conocido y estando bien cocinado, se han cogido todas las entradas que cumplieran dichos requisitos y se han calculado la media y la desviación típica.

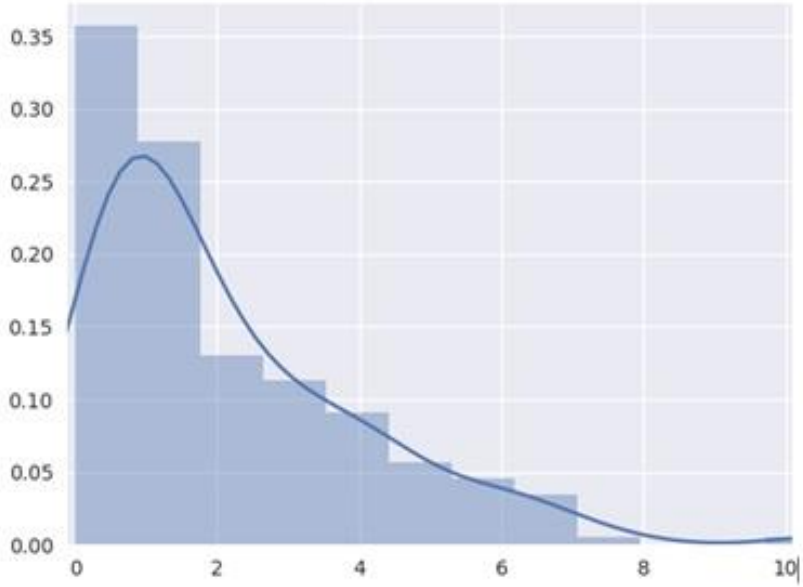
A continuación mostramos un ejemplo de cada distribución usada tanto para el perrito como para la ensalda

Perrito caliente



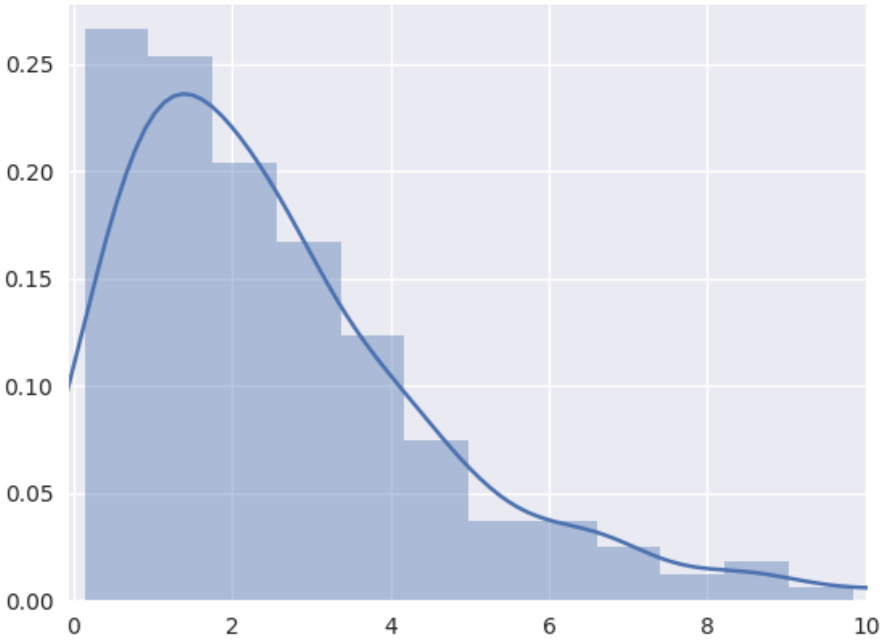
$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Figura 4.2-5: Distribución normal de μ=5.5 y σ^2=1.8



$$f(x;\lambda) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0, \\ 0 & x < 0. \end{cases}$$

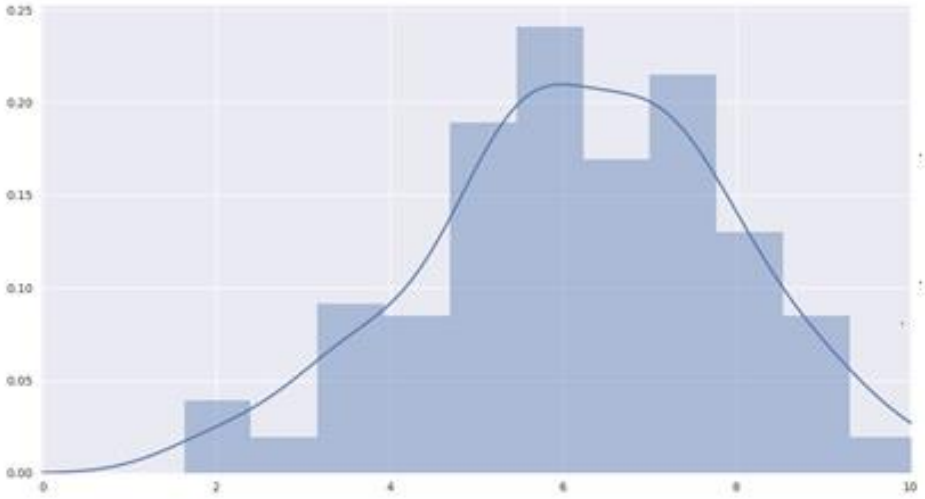
Figura 4.2-6: Distribución exponencial de λ=2



$$f(x;k) = \begin{cases} \frac{1}{2^{k/2}\Gamma(k/2)} x^{(k/2)-1} e^{-x/2} & \text{para } x > 0, \\ 0 & \text{para } x \leq 0 \end{cases}$$

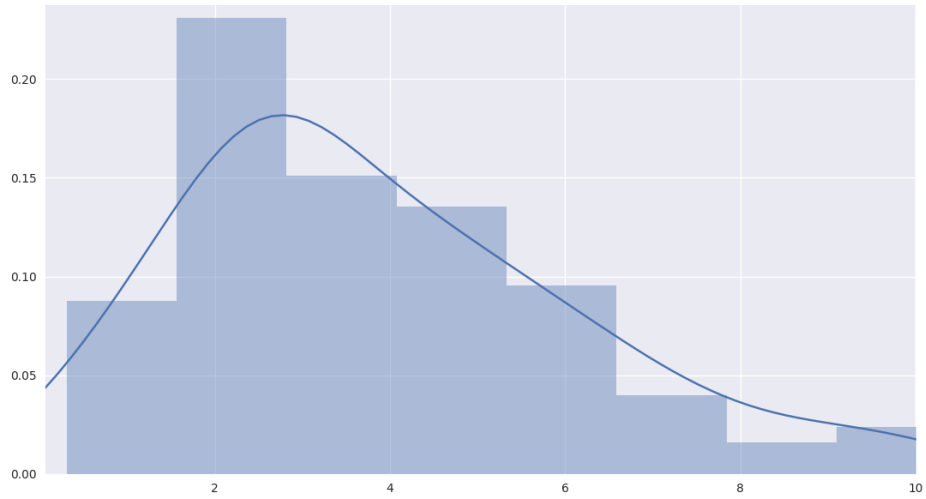
Figura 4.2-7: Distribución chis de k=2.5

Ensalada



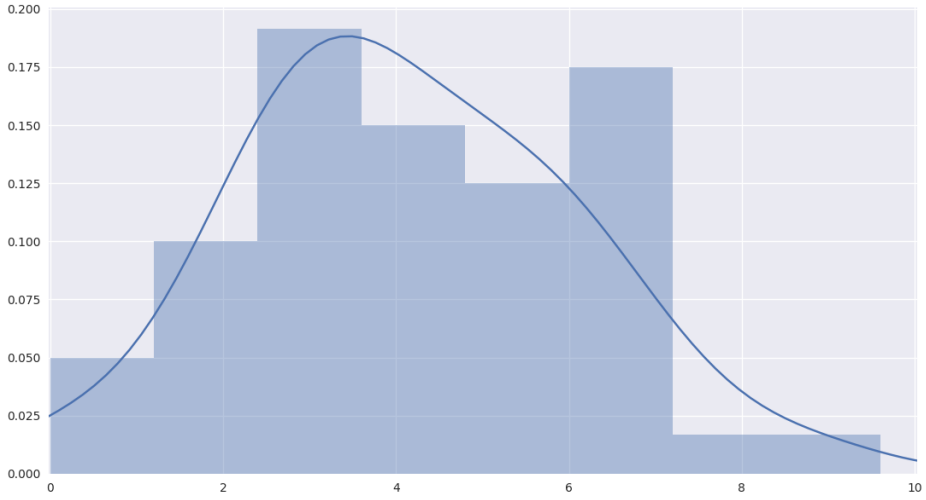
$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Figura 4.2-8: Distribución normal de μ=8.5 y σ^2=0.95



$$f(x) = \lambda e^{-\lambda x} \frac{(\lambda x)^{k-1}}{\Gamma(k)}$$

Figura 4.2-9: Distribución gamma de λ=2 y k=2



$$f(k, \lambda) = \frac{e^{-\lambda} \lambda^k}{k!}$$

Figura 4.2-10: Distribución Poisson de λ=4

4.3 Entrenamiento del modelo

Con los dos conjuntos de datos que creamos previamente, el siguiente paso fue entrenar un modelo que se ajustara a los datos.

En este punto, no había ninguna idea preconcebida sobre usar específicamente un modelo, así que se decidió comenzar probando el modelo SVR.

Para entender cómo funciona una SVR primero debemos abordar las SVN. Imaginemos un problema de clasificación en el que tenemos dos clases, la clase círculo y la clase cuadrado. Para poder clasificar correctamente qué puntos serán círculos o cuales serán cuadrados, bastaría con definir un hiperplano que separe ambas zona de tal forma que si el punto a clasificar está en uno de los lados de la frontera sea círculo, y si está en el otro,

sea cuadrado. Como se puede observar en la figura 4.3-1 hay infinitos hiperplanos que permitan hacer esta distinción, por lo que debemos ser capaces de saber cuál es el mejor de ellos.

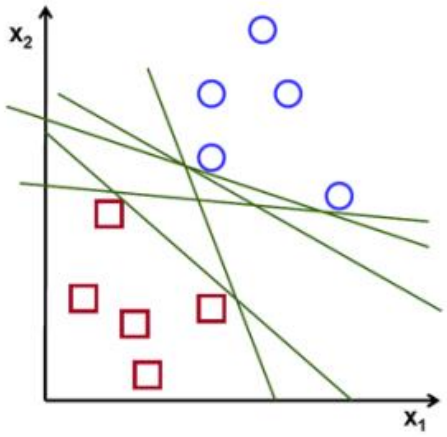


Figura 4.3-1: Posibles fronteras de separación

Para elegir el hiperplano que mejor clasifique los puntos, debemos recurrir al concepto de *margen* que es la distancia mínima que hay entre el hiperplano y el punto que más cerca se encuentra de cada clase. Siendo *w* un plano tal que $w \equiv Ax + By + Cz + D = 0$ y *x'* un punto tal que $x' (x_0, y_0, z_0)$, por geometría, la distancia entre el hiperplano *w* y un ejemplo *x'* viene dada por

$$d(x', w) = \frac{|Ax_0 + By_0 + Cz_0 + D|}{\sqrt{A^2 + B^2 + C^2}} \quad (4)$$

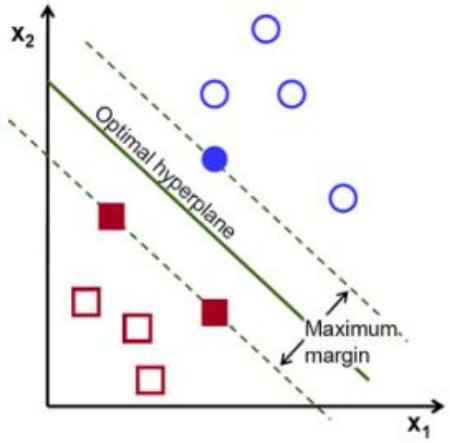


Figura 4.3-2: Margen de distancia de hiperplano a cada clase

Sin embargo, no todos los casos son separables como los casos que hemos visto en la figura 4.3-2 sino que lo habitual es que el caso no sea lineal, es decir, no siempre tenemos un problema tan sencillo como clasificar un punto dentro de dos posibles clases que son separables, sino que necesitamos clasificar puntos que en dos dimensiones no se pueden diferenciar. Para ello, se transforma el problema que teníamos a una dimensión en la que el problema sí sea separable como muestra la figura 4.3-3

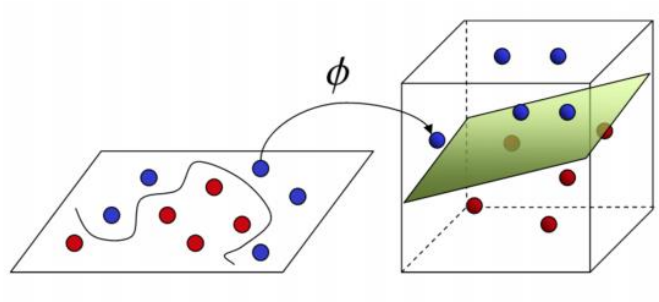


Figura 4.3-3: Transformación de dimensión

Dado que esta solución está demostrado que tiene un buen rendimiento, surge la SVR cuya idea es seleccionar un hiperplano, al igual que en la SVN pero en este caso de tipo regresor. Este hiperplano regresor se debe seleccionar en función del que mejor se ajuste a los datos de entrenamiento que tengamos, para de este modo, no sólo clasificar puntos, sino variables reales, que es el caso que nos atañe con las notas que obtiene cada plato de comida. La función de predicción de la SVR viene dada por

$$f(x) = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) \langle x_i, x \rangle + b \tag{5}$$

donde α_i y α_i^* son variables asociadas a las restricciones, x_i y x es el ejemplo a clasificar y b es una constante calculada con las condiciones de complementariedad de Karush-Kuhn-Tucker [14]

Así, tal y como hemos mencionado antes, se ha utilizado la librería de *SVR* de *Python* que permite usar este modelo. Esta librería recibe dos parámetros: *C* y *gamma* que permiten ajustar el modelo para la forma más óptima posible. Para determinar los valores que mejor ajustaban nuestro modelo para cada uno de los dos experimentos, se ha realizado una búsqueda en rejilla en función de unos valores que expertos en la materia proporcionaron. Por un lado, para el valor de *C* se usó el vector [0.1, 1.0, 10.0, 100.0] y para el valor de gamma, se usó el vector [0.01, 0.1, 1.0, 10.0]. Con esto, la rejilla quedó como muestra la figura 4.3-4

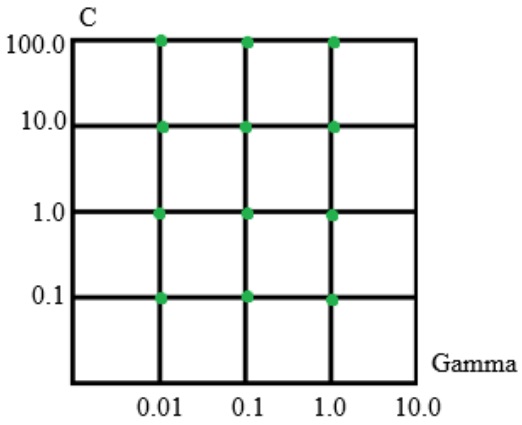


Figura 4.3-4: Rejilla de búsqueda para SVR

Así, probamos cada uno de las combinaciones de valores de la rejilla. Cabe destacar, que para establecer el acierto de cada una de las combinaciones, lo hicimos calculando el error cuadrático medio de la nota que predecía nuestro modelo respecto de la nota que realmente el plato tenía como muestra la ecuación (6)

$$ECM = \frac{1}{n}\sum_{i=1}^{n}(Y_i - X_i)^2 \quad (6)$$

donde Y_i es el vector de las predicciones, X_i es el vector de resultados reales y n es el número de ejemplos que estamos evaluando.

En la tabla 4.3-1 y 4.3-2 mostramos el error cuadrático medio que obtuvieron cada una de las parejas de valores para cada uno de los platos:

Perrito		**Gamma**			
		0.01	**0.1**	**1.0**	**10.0**
C	**0.1**	3.00	4.03	5.56	5.8
	1.0	2.56	2.86	4.38	5.37
	10.0	2.76	3.05	3.79	4.90
	100.0	3.43	3.80	3.77	4.90

Tabla 4.3-1: Resultados de búsqueda para perrito

Ensalada		**Gamma**			
		0.01	**0.1**	**1.0**	**10.0**
C	**0.1**	2.80	3.35	7.02	8.40
	1.0	2.60	2.72	4.19	6.95
	10.0	2.65	3.22	3.93	6.15
	100.0	3.28	3.88	3.93	6.07

Tabla 4.3-2: Resultados de búsqueda para ensalada César

Como se puede observar, en ambos casos la mejor combinación posible es con *C*=1.0 y *Gamma* = 0.01.

Finalmente, también hay que mencionar que para tratar de ajustar el modelo de la mejor forma posible, se hizo una validación cruzada de 10 iteraciones. Esto consiste en separar el conjunto de datos en dos partes, una parte para entrenar el modelo y otra parte (usualmente menor) para validar el modelo y ver cuán bueno es prediciendo.

En cada iteración se varían estas divisiones de tal forma que para cada iteración el conjunto de entrenamiento y de prueba, no sean el mismo, tal como se puede observar en la figura 4.3-5.

Con esto lo que conseguimos, es saber el error medio real del modelo, pues si solo separáramos en entrenamiento y test una vez, pudiera darse la casuística de que en esa división concreta el conjunto de entrenamiento fuera excepcionalmente bueno para la posterior clasificación o a la inversa, que fuera excepcionalmente malo, por lo que no serían resultados que realmente nos dieran información útil.

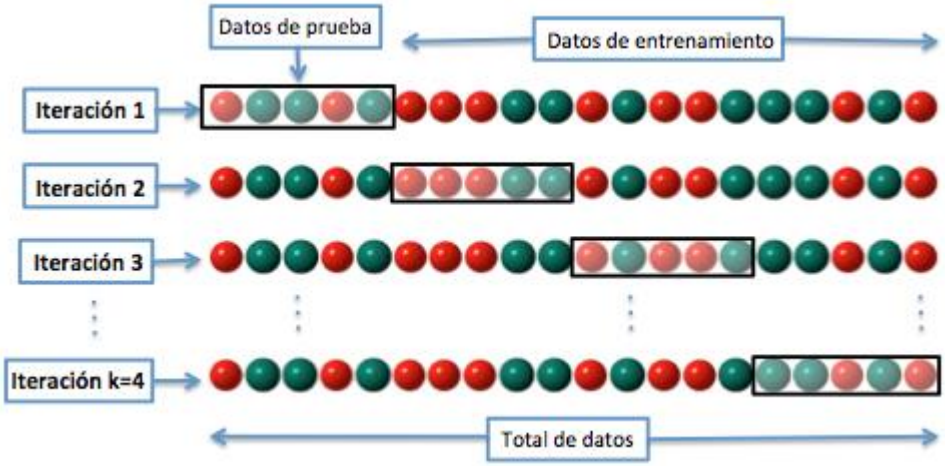


Figura 4.3-5: Validación cruzada

Con esto ya tendríamos nuestra SVR entrenada. Como los resultados con este modelo fueron satisfactorios, no proseguimos a buscar más modelos, por lo que el siguiente paso en el proyecto fue la optimización Bayesiana.

4.4 Optimización Bayesiana

Todos los algoritmos de aprendizaje automático tienen un conjunto propio de parámetros que deben ser ajustados de manera óptima. En nuestro caso, estos parámetros como hemos visto, son los ingredientes y las herramientas para cocinarlos.

El cambio de alguno de estos parámetros o del número de los mismos, puede impactar en el rendimiento del sistema.

La elección de estos parámetros, comúnmente se ha realizado por humanos expertos en el problema que se esté tratando, que usualmente consiste.

Como hemos podido ver en el estado del arte, la optimización Bayesiana es un algoritmo de búsqueda que nos permite obtener la combinación óptima de parámetros para un problema determinado siendo una buena alternativa a la búsqueda en rejilla que adolece de la maldición de la dimensionalidad.

Más concretamente, la optimización Bayesiana es especialmente útil en casos en los que tenemos que optimizar objetivos de los cuales no sabemos su expresión analítica, que son caros de evaluar y que son potencialmente ruidosos, como es el problema del presente trabajo.

Esto nos da lugar a que tendremos una serie de puntos que habremos descubierto, pero sin saber cómo es realmente la función que estamos estudiando. La figura 4.4-1 representa esto de una forma muy clara; tenemos 5 puntos que hemos descubierto, pero sin embargo, al no tener la función real, cualquiera de las tres funciones representadas (verde, azul y roja), podrían ser representaciones válidas.

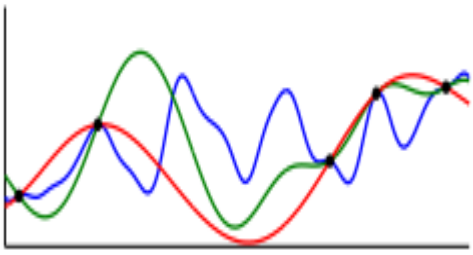


Figura 4.4-1: Ejemplo en función de los hiperparámetros

Consideremos una función de caja negra $f(\ .\ )$ que tiene una evaluación ruidosa del tipo $y_i = f(x_i) + \epsilon_i$ siendo ϵ_i el término que representa el ruido. La optimización Bayesiana es muy útil en reducir el número de evaluaciones de la función objetivo, necesarias para resolver el problema de optimización.

En cada iteración del proceso de optimización, el método ajusta un modelo probabilístico (usualmente se usa un proceso Gaussiano) a las observaciones de la función objetivo $\{yi\}_{i=1}^{t-1}$.

La incertidumbre sobre la función objetivo creada por el proceso Gaussiano es usada por una función llamada función de adquisición $\alpha(\cdot)$ cuyo valor en cada posible punto de entrada indica la utilidad esperada por la función objetivo $f(\cdot)$. El criterio que sigue para saber qué punto es el próximo a evaluar en la función objetivo $f(\cdot)$ es aquel que maximice el valor de $\alpha(\cdot)$. Es importante aclarar que $\alpha(\cdot)$ solo depende del modelo probabilístico y que las evaluaciones que se hacen en este punto tienen un coste muy bajo por lo que esta función puede ser maximizada de una manera muy rápida usando técnicas de optimización más usuales.

Este proceso se repite hasta que se tienen suficientes datos de la función objetivo, momento en el cual el proceso Gaussiano predice un $f(\cdot)$ óptimo para encontrar la solución al problema de optimización. En la figura 4.4-2 se puede observar el algoritmo de optimización Bayesiana de una función de caja negra [12].

for $t = 1, 2, 3, \ldots,$ *max_steps* **do**
  1: Find the next point to evaluate by optimizing the acquisition function: $\mathbf{x}_t = \arg\max_{\mathbf{x}} \alpha(\mathbf{x}|\mathcal{D}_{1:t-1})$.
  2: Evaluate the black-box objective $f(\cdot)$ at $\mathbf{x}_t$: $y_t = f(\mathbf{x}_t) + \epsilon_t$.
  3: Augment the observed data $\mathcal{D}_{1:t} = \mathcal{D}_{1:t-1} \bigcup \{\mathbf{x}_t, y_t\}$.
  4: Update the Gaussian process model using $\mathcal{D}_{1:t}$.
end
Result: Optimize the mean of the Gaussian process to find the solution.

Figura 4.4-2: Algoritmo de optimización Bayesiana

Como hemos dicho antes, el éxito de la optimización Bayesiana está en que evaluar la función de adquisición $\alpha(\cdot)$ es mucho más barato que evaluar la función objetivo $f(\cdot)$. Esto es así debido a que la función de adquisición solo depende de que el proceso Gaussiano prediga sobre la función objetivo en un único punto *x*.

Como hemos visto, la función de adquisición es un punto determinante en el algoritmo de la optimización Bayesiana. Un ejemplo de función de adquisición es la *mejora esperada* (*EI: Expected Improvement*) que se obtiene como el punto que se cree que mayor valor va a tener en una función de utilidad $u(y_i) = max(0, v\text{-}y_i)$ donde $v = min (\{yi\}_{i=1}^{t-1})$ es el mejor valor observado hasta el momento, es decir, la *mejora esperada* mide cómo puede mejorar la mejor solución hasta el momento evaluando cada uno de los puntos candidatos.

Así, para aplicar este algoritmo a nuestro problema, usamos la herramienta *Speartmint* que es de código abierto, y que permite obtener la optimización de una manera sencilla. Basta con darle un fichero de configuración de tipo *.json* que se usa principalmente para especificar las variables que tiene nuestro problema, el tipo del que son, y el rango que tienen, tal y como se indica en la figura 4.4-3

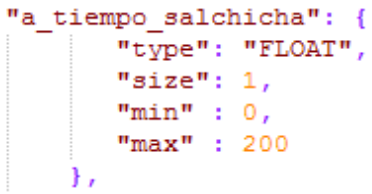


Figura 4.4-3: Ejemplo fichero .json

y un fichero *Python* que se encargue de cargar el *.json* cargado anteriormente y llamar a *Spearmint*.

Con esto obtuvimos unos resultados que se analizan en el siguiente capítulo.

5 Experimentos

En esta sección describiremos la ejecución de los experimentos que se han realizado y veremos los resultados que se han obtenido, comparándolos con los resultados esperados y analizando a qué se deben.

Los resultados que aquí exponemos mediante gráficas son fruto de ejecutar 100 veces Spearmint, y los histogramas corresponden a las recomendaciones de la última iteración, que son el mejor resultado observado de todas las iteraciones.

5.1 Perrito

En primer lugar analicemos qué resultados se obtuvieron con el perrito. Para poder saber si la optimización Bayesiana recomienda unos rangos para las variables que concuerden con las reglas que obtuvimos por el conocimiento experto, es necesario analizarlas de manera independiente.

Pan

La figura 5.1-1 muestra la frecuencia de los tiempos que más se han recomendado. Como se puede observar, el tiempo que mejor resultados ha obtenido es el que oscila entre [100,115] segundos. Si comparamos esto con nuestro conocimiento experto, vemos que hay concordancia, pues se establecía un rango entre [100,110] segundos

Figura 5.1-1: Resultado de la variable *tiempo* para *pan* de perrito

Salchicha

En cuanto a la salchicha, tenemos que analizar tanto el tiempo empleado para cocinarla, como el lugar elegido para hacerlo.

Según muestra la figura 5.1-2 el tiempo más recomendado ha sido de [45,55] segundos con una frecuencia del 62%. Al compararlo con las reglas que se establecieron, vemos que de nuevo se obtienen resultados similares, pues el conocimiento experto tasaba el tiempo idóneo en [35,45] segundos.

En cuanto al lugar elegido para cocinar la salchicha, la figura 5.1-3 muestra que más del 85% de las veces se ha recomendado usar la sartén que concuerda plenamente con el conocimiento experto

Figura 5.1-2: Resultado de la variable *tiempo* para *salchicha* de perrito

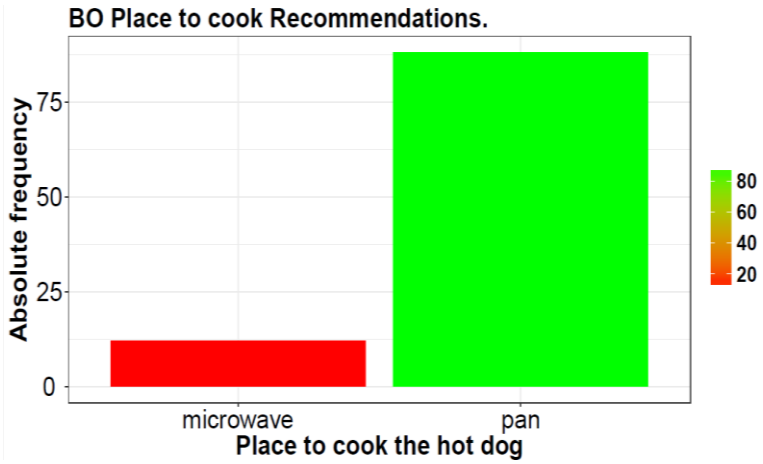


Figura 5.1-3: Resultado de la variable *lugar* para *salchicha* de perrito

Salsas

En cuanto a las salsas, las figuras 5.1-4, 5.1-5 y 5.1-6 y muestran la recomendación de mostaza, salsa barbacoa y mayonesa respectivamente.

Como se ve, la optimización Bayesiana ha optado por recomendar que el perrito no lleve ninguna salsa. En este caso, el resultado obtenido no concuerda con lo marcado por el conocimiento experto.

Esto puede deberse a que en las reglas, para tener un buen resultado respecto a las salsas, había que lograr una combinación muy concreta, es decir, las posibles combinaciones de cantidades de cada salsa son tan grandes, que la probabilidad de encontrar una combinación buena dentro de ese espacio, es muy baja.

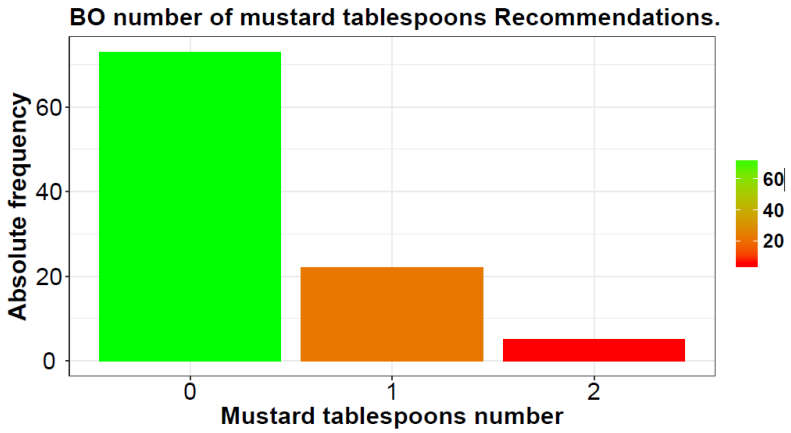


Figura 5.1-4: Resultado de la variable *mostaza* para perrito

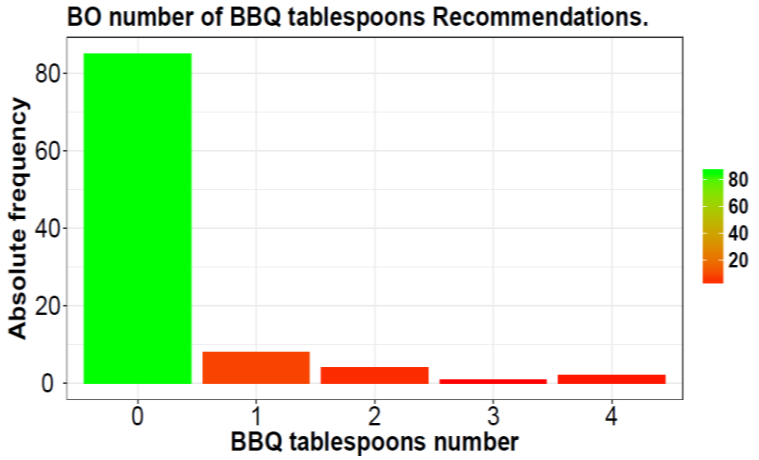


Figura 5.1-5: Resultado de la variable *barbacoa* para perrito

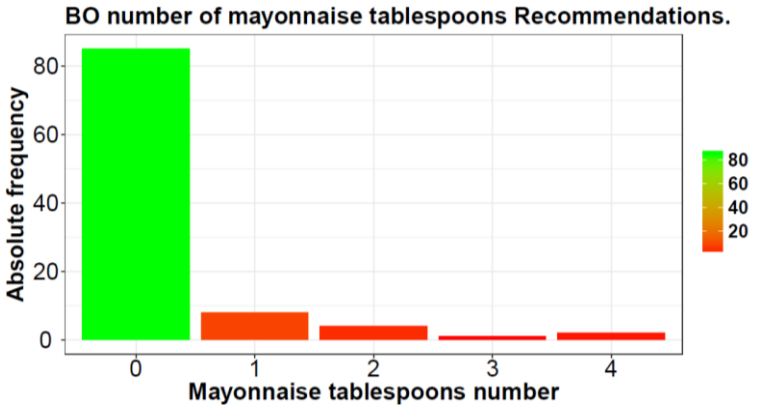


Figura 5.1-6: Resultado de la variable *mayonesa* para perrito

Finalmente, en la figura 5.1-7 podemos ver la evaluación de la eficiencia de la optimización Bayesiana respecto a la búsqueda aleatoria y respecto al conocimiento experto.

Se puede ver cómo lógicamente el conocimiento experto no sufre un aumento de la nota aunque se aumenten las evaluaciones, pues se basa siempre en las reglas que ya tiene.
Sin embargo, tanto la optmización Bayesiana como la búsqueda aletoria consiguen mejorar su evaluación en cada iteración, siendo la optimización Bayesiana notablemente mejor. Esto se debe a que la búsqueda aleatoria necesita muchos puntos a estudiar en grandes dimensiones mientras que los procesos gaussianos que usa la optimización Bayesiana, requiere de menos puntos de media

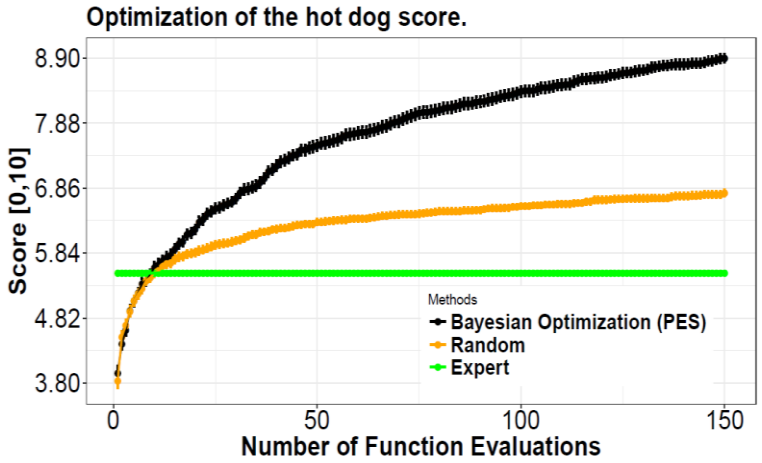


Figura 5.1-7: Comparación de optimización Bayesiana para perrito

5.2 Ensalada

Al igual que con el perrito, vamos a analizar de manera independiente cada variable de la ensalada César.

Pan

Las figuras 5.2-1 y 5.2-2 muestran respectivamente el tiempo empleado para freír el pan para hacer picatostes así como la potencia de la empleada en la vitrocerámica.
Como se puede ver, los resultados obtenidos son [12,15] segundos con la potencia de la vitrocerámica al 9.

La recomendación del conocimiento experto se estableció en [10,15] segundos con potencia de la vitrocerámica al 9, por lo que los resultados concuerdan con lo esperado.

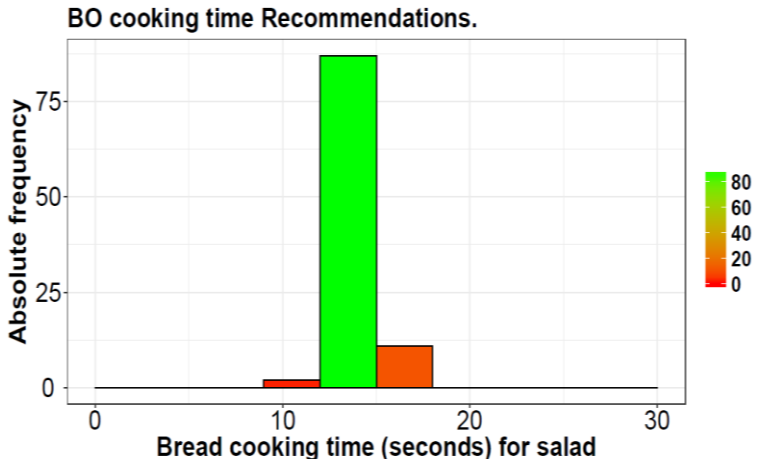


Figura 5.2-1: Resultado de la variable *tiempo* para *pan* de ensalada

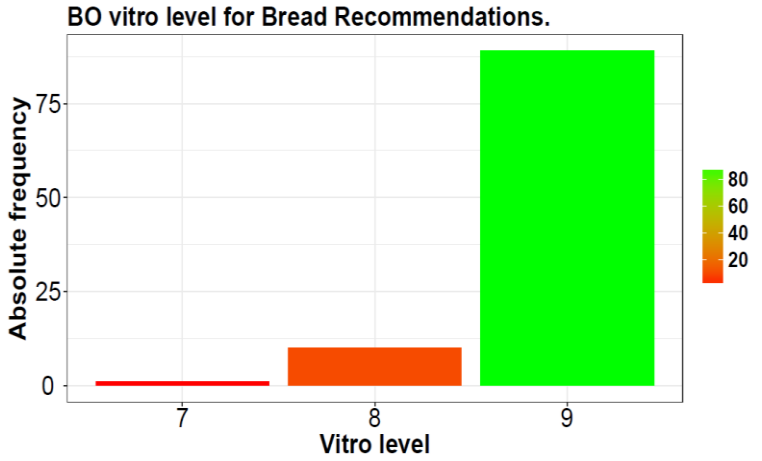


Figura 5.2-2: Resultado de la variable *vitrocerámica* para *pan* de ensalada

Pollo

La recomendación para cocinar el pollo se muestra en las figuras 5.2-3 y 5.2-4 donde se puede observar que tenemos unos valores de [70, 100] segundos con la vitrocerámica a la potencia 8. Los resultados del conocimiento experto, fueron [90,115] segundos con la potencia al [8,9].

Los resultados aunque no se acercan tanto al conocimiento experto como en sucede con la variable pan, dan valores que obtienen una buena puntuación. Cabe destacar que aunque el mayor porcentaje (alrededor de 60%) de recomendación de potencia de vitrocerámica sea con ella al 8, un 35% de las veces es con la misma al 9, por lo que la relación de tiempo empleado en cocinar el pollo con la potencia de la vitrocerámica, sí se ajusta a valores con buenos resultados.

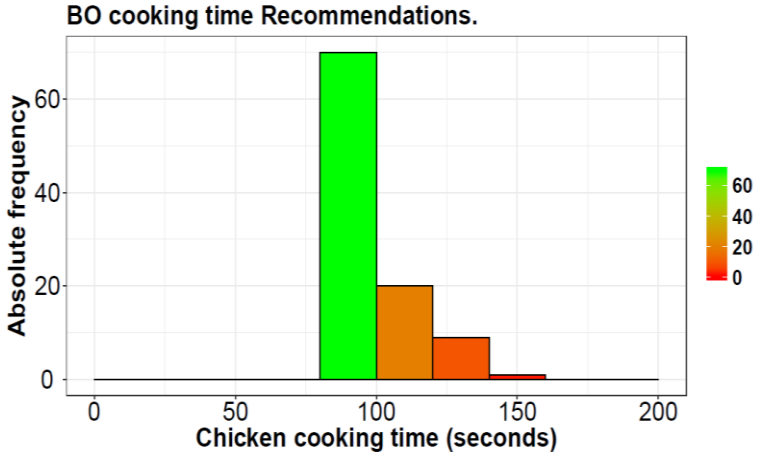


Figura 5.2-3: Resultado de la variable *tiempo* para *pollo* de ensalada

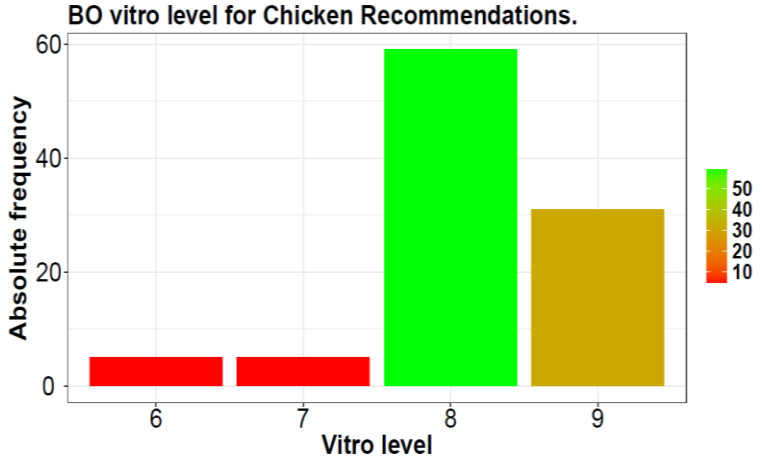


Figura 5.2-4: Resultado de la variable *vitrocerámica* para pollo ensalada

Lechuga

La figura 5.2-5 muestra como la optimización Bayesiana opta por las marcas de lechuga de *Curro* y *Giante Verde* que coincide con las recomendaciones del conocimiento experto

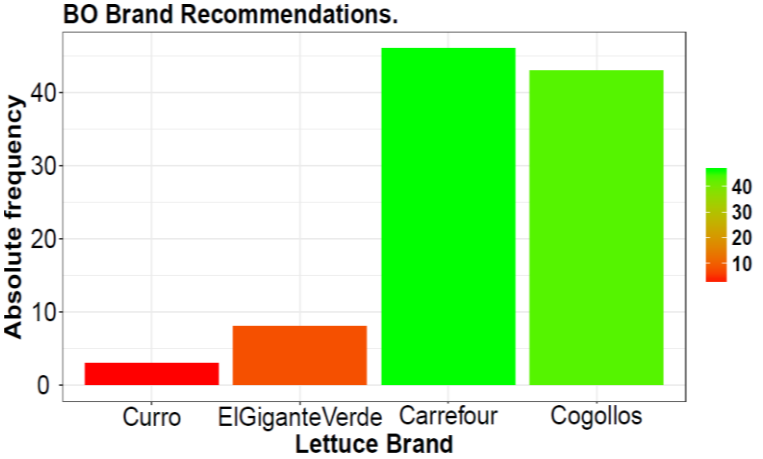


Figura 5.2-5: Resultado de la variable *lechuga* para ensalada

Salsa

En la figura 5.2-6 se puede ver que al igual que recomienda el conocimiento experto, la optimización Bayesiana opta por que los mejores resultados se dan con la salsa César de marca Ybarra

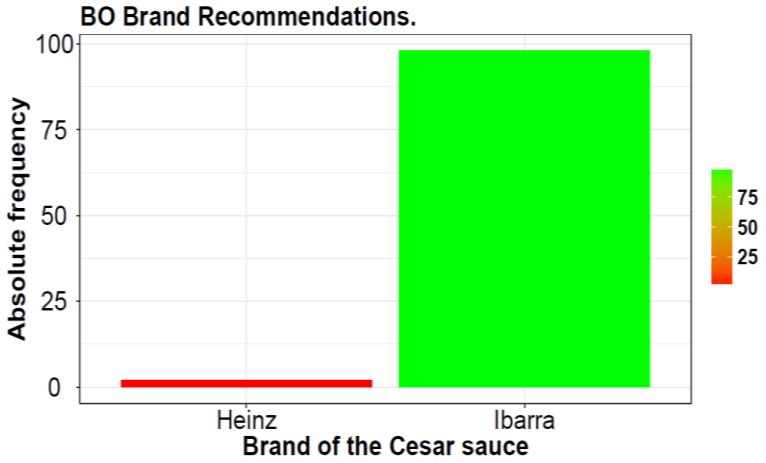


Figura 5.2-6: Resultado de la variable *salsa* para ensalada

Finalmente, en la figura 5.2-7 se muestra una comparación de la optimización Bayesiana, la busqueda aleatoria y el conocimiento expertoy, al igual que con el perrito, el conocimiento experto no sufre ninguna mejora, pues tiene los datos de partida y no hace nada para mejorar de manera iterativa.

En cuanto a la optmización Bayesiana y la búsqueda aletoria consiguen mejorar sus resultados gradualmente, teniendo la optimización Bayesiana un coeficiente de arendizaje mayor, pues como se observa, en el mismo número de evaluaciones, logra una nota mejor (8.8 de la optimización Bayesiana por aproximadamente 7.6 de la búsqueda aleatoria). Esto se debe a que la búsqueda aleatoria necesita muchos puntos a estudiar en grandes dimensiones mientras que los procesos gaussianos que usa la optimización Bayesiana, requiere de menos puntos de media.

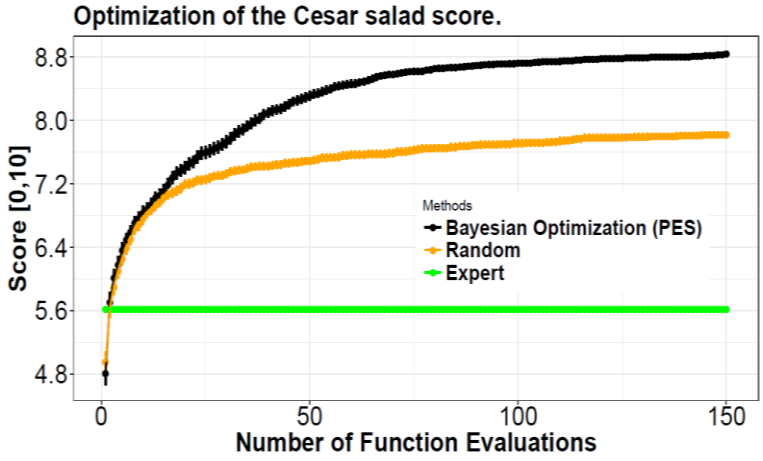


Figura 5.2-7: Comparación de optimización Bayesiana para ensalada

6 Conclusiones y trabajo futuro

6.1 Conclusiones

A lo largo de la presente memoria, hemos podido comprobar el gran campo de inteligencia artifical que es el aprendizaje automático y hemos podido sacar algunas conclusiones:

- El campo de la inteligencia artificial es un ámbito aún con grandes aspectos por descubrir y en el que en muchas ocasiones se piensan que tenemos unos conocimientos que en realidad no tenemos. La inteligencia artificial no es más que una serie de algoritmias que permiten automatizar procesos cada vez más complejos, pero dista mucho del pensamiento que tiene la mayor parte de la humanidad al respecto, como pudiera ser en este caso, la creación de un robot con manos y piernas que fuera capaz de cocinarnos platos sabrosos.

- Por otra parte, si se toma la actitud adecuada, el aprendizaje automático puede ser de gran utilidad. En este caso hemos sido capaces con unos recursos muy limitados de conseguir unos resultados realmente satisfactorios. Esto no es más que un pequeño ejemplo del potencial explotable que hay.

- La cocina precisa de un estudio minucioso con equipos multidisciplinares, pues es necesario aglutinar a perfiles técnicos, junto con expertos del mundo culinario

6.2 Trabajo futuro

En este proyecto hemos podido ver la gran utilidad del sistema creado. Sin embargo, la idea es que este sistema crezca para que pueda extrapolarse a recetas de mayor complejidad.

Por tanto, a continuación mostramos algunas de las posibles mejoras que se pretenden realizar en el futuro:

- Más datos reales. En este trabajo de fin de grado los recursos con los que se han trabajado han sido muy limitados, lo cual ha llevado a la necesidad de hacer un aumento *ficticio* de los mismos. Un buen punto de mejora sería poder contar con unos conjuntos de datos mayores, tomando también las notas de un número mayor de personas, no solo de tres como ha sido el caso.

- Tomar medidas más exactas. Debido al alcance de este proyecto, había ciertas variables que no podían llevarse con toda la exactitud que sería recomendable. Por ejemplo, a la hora de tostar el pan o freír un filete, en lugar de establecer la potencia de la vitocerámica, una medida mejor sería poder establecer que el aceita tuviera una temperatura determinada midiéndolo con un termómetro. Otro ejemplo sería

que en lugar de medir las salsas en cucharadas, medirlas en gramos. Con esto podríamos realizar recetas más rigurosas.

- Restricciones y multiobjetivo. En nuestro caso únicamente hemos tenido un objetivo en el experimento (el sabor) y no hemos tenido ninguna restricción. Sin embargo, en la vida real, una comida no se mide sólo por su sabor, sino que también hay otras componentes como la presentación. Así mismo también se podría introducir evaluaciones con restricciones, como por ejemplo que un plato de comida no sobrepasara un precio por plato determinado o que el tiempo empleado en cocinarlo no sobrepasara cierto umbral.

- Procesador de lenguaje natural. Como se ha observado, nosotros hemos partido de una serie de reglas en lenguaje natural que nos dio el conocimiento experto. Un módulo de mejora interesante sería incorporar un procesador de lenguaje natural que permitiera captar estas reglas para que de manera automática pudiera realizar el resto de proceso de optimización.

Referencias

[1] Andrés González. "¿Qué es machine learning?" http://cleverdata.io/que-es-machine-learning-big-data/

[2] Kirstine Smith. 1918. "On the standard deviation of adjusted and interpolated values of an observed polynomial function and its constants and the guidance they give towards a proper choice of the distribution of observations"

[3] H. J. Kushner. "A new method for locating the maximum point of an arbitrary multipeak curve in the presence of noise. Journal of Basic Engineering"

[4] Brochu Eric, Nando D. Freitas, Abhijeet Ghosh "Active Preference Learning with Discrete Choice Data"

[5] Prasoon Goyal. "Machine Learning: How does grid search work?". https://www.quora.com/Machine-Learning-How-does-grid-search-work

[6] James Bergstra , Yoshua Bengio. "Random Search for Hyper-Parameter Optimization" http://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf

[7] Wikipedia. "Algoritmo genético" https://es.wikipedia.org/wiki/Algoritmo_gen%C3%A9tico

[8] "Spearmint: Bayesian optimization codebase" http://github.com/HIPS/Spearmint/

[9] Package 'rBayesianOptimization' https://cran.r-project.org/web/packages/rBayesianOptimization/rBayesianOptimization.pdf

[10] Richard Brandt. "Chef Watson has arrived and is ready to help you cook". https://www.ibm.com/blogs/watson/2016/01/chef-watson-has-arrived-and-is-ready-to-help-you-cook/

[11] Greg Kochanski, Daniel Golovin, John Karro, Benjamin Solnik, Subhodeep Moitra, and D. Sculley "Bayesian Optimization for a Better Dessert"

[12] Numpy http://www.numpy.org/

[13] Pedro Domingos. “Why Does Bagging Work? A Bayesian Account and its implications”
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.40.1298&rep=rep1&type=pdf

[14] David G Luenberger y Manuel López Mateos. Programación lineal y no lineal. Number 90C05 LUEp. Addison-Wesley Iberoamericana, 1989.

[15] Laura Cornejo, Eduardo Garrido, Daniel Hernández, Sancho Salcedo “Bayesian Optimization of a Hybird System for Robust Ocean Wave Features Prediction”
https://www.researchgate.net/profile/Laura_Cornejo_Bueno/publication/319857422_Bayesian_Optimization_of_a_Hybrid_System_for_Robust_Ocean_Wave_Features_Prediction/links/5a0ac1830f7e9bb949f99486/Bayesian-Optimization-of-a-Hybrid-System-for-Robust-Ocean-Wave-Features-Prediction.pdf

Glosario

IA Inteligencia Artificial
OB Optimización Bayesiana
SVM Máquina de Vector Soporte
SVR Regresión de Vector Soporte